ISSN ONLINE (2395-1680) ICTACT JOURNAL ON MICROELECTRONICS, FEBRUARY 2015, VOLUME: 01, ISSUE: 01
DOI: 10.21917/ijme.2015.0008

ON-DEMAND CHECKPOINT AND RECOVERY SCHEME FOR WIRELESS SENSOR NETWORKS

N. Nithilan[1] and A. Pravin Renold[2]
TIFAC-CORE in Pervasive Computing Technologies, Velammal Engineering College, India
E-mail: [1]nithilan16@gmail.com, [2]pravinrenold.tifac@velammal.edu.in

Abstract

In future, WSN (Wireless Sensor Network) which consists of n number of tiny wireless embedded systems will be an indispensable part of our daily life. Fault tolerance is an imported technique which has to be implemented in sensor networks to avoid catastrophic events in the network; failure in the network may lead to wastage of resources. Fault tolerance is the property that enables a system to continue operate properly in the event of failure of (or one or more faults within) some of its components. An improved version of checkpointing called on-demand checkpoint and rollback recovery scheme is implemented in this paper. The cluster head collects data from its member nodes and aggregates the collected data. To improve reliability and reduce recovery latency, our method uses coordinating checkpoint, non blocking checkpoint to make the checkpoint taken consistent in nature. In the proposed scheme, backup nodes monitor and checkpoint the current state of the cluster head on demand. Experimental comparisons with CTP (Collection Tree Protocol) and demand checkpoint with CTP shows time and Packet Delivery Ratio values are identical saying checkpointing does not affect the WSN performance due to inclusion of checkpoint. The proposed scheme is designed and implemented in Contiki OS using sensor network simulator COOJA.

Keywords:

Fault Tolerance, On-Demand Checkpointing, Rollback Recovery, Contiki OS, COOJA Simulator

1. INTRODUCTION

Wireless Sensor Network (WSN) refers to a group of spatially dispersed and dedicated sensors for monitoring and recording the physical conditions of the environment and organizing the collected data at a central location. WSNs measure environmental conditions such as temperature, sound, pollution levels, humidity, wind speed, direction and pressure. Each sensor node comprises components like controller unit for processing all the relevant data and capable of executing arbitrary code, sensors and actuator unit used to observe or control physical parameter of the environment, memory unit to store program and data [1].

Wireless sensor nodes are also called as motes which provide bridge between the real physical and virtual world. In real scenario more than thousand of motes are deployed in the real physical world to monitor an area under study [1]. WSN has a wide range application in various fields such as process management, health care monitoring, environment monitoring, water quality monitoring and industrial monitoring.

Important challenges in WSN are energy consumption, localization, data gathering, reliability, fault tolerance, scalability and security. All human made systems are vulnerable to fault. With each new emerging technology, we can’t predict whether it will continue to work through out or till prescribed time. These technologies are liable to failure and it is hard to predict when a failure will happen. WSN is designed to minimize the occurrence of faults, yet it is understood that fault are bound to occur and caused by, wear and tear due to environmental condition, or by some unavoidable conditions, design or usage errors. A fault refers to an abnormal physical state of a sensor node which may be caused by factors like change in humidity, temperature, power surge, electromagnetic radiations, design or installation errors and ageing. The faults can be characterized as [2] crash fault, omission fault, timing fault, fail-stop fault and byzantine fault. The most basic fault classes such as crash, fail-stop, omission, and timing failures, are problems that occur and detected in the time domain. The fault classes like incorrect computation and Byzantine faults occur in data domains.

For many systems handling of faults in terms of diagnostic, repair and maintenance occurs offline i.e. outside the system. A failure of the system may be annoying, but are generally neither life threatening nor safety critical and the cost of the system being down or behaving erroneous is not prohibitive, in some system incorrect behaviour carries serious consequences.

1.1 CHECKPOINTING AND ROLLBACK RECOVERY

Fault tolerance in WSN is an important challenge which should be solved to make a network sustainable and error free. Unfortunately fault tolerance is not considered as an important challenge by researchers and work done on fault tolerance and recovery are very limited [3]. If faults are not considered as a serious one, the erroneous system may lead to many losses, including data, cost involved, life involved in a critical event monitoring system like patient monitoring system. Since motes involved in the system are large in number and in interlinked fashion, fault in single mote can propagate and affect other nodes involved in the system and consequently make the system inconsistent. Checkpoint and rollback recovery is one important method implemented to make a distributed system i.e. WSN, a fault tolerable or recoverable system. Checkpoint implemented does not consider state loss as a serious problem. So as a result the state embedded in the sensor is lost which may result in data loss [3].

A checkpoint is a record of certain consistent state of a process. When a distributed system fails, it can be recovered to a consistent state by the checkpoints stored in the stable storages or non-volatile storage. For a general checkpointing strategy, it mainly takes recovery overhead and storage overhead into account. Checkpointing aims to provide recovery capability for application programs.

Fault tolerance can be achieved in various forms, one technique among them is called as recovery scheme, i.e. recovering the faulty state to a consistent state, and recovery can be of two type's forward and backward recovery. Forward recovery means moving the present faulty state to a safer or error free state which is completely new to the system, backward recovery means moving the faulty state to an already known error free state.

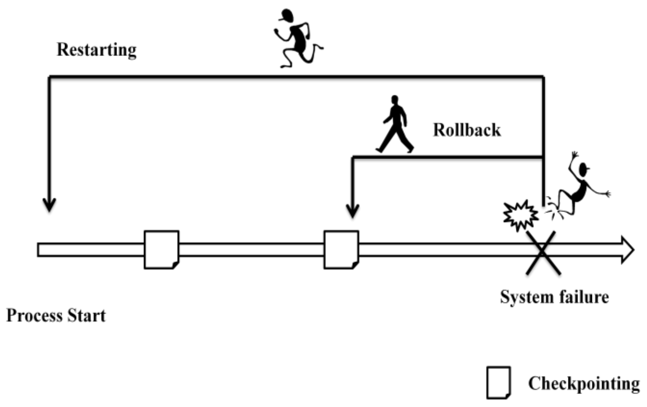


Fig.1. Checkpointing and rollback recovery

The working of checkpointing and rollback recovery is specified in Fig.1, due to some unavoidable reasons the system fails at a certain point, in conventional systems, the system will be restarted from the first there by taking large time for coming back to the normal working, but in our system due to checkpointing the system will be recovered from most recent known error free state by the concept called rollback recovery.

Rollback recovery is briefly classified [8] into checkpoint based and log based system. The checkpoint based recovery can be achieved in three forms namely: uncoordinated, communication induced or coordinated checkpoints. Further, coordinated can be classified as blocking, non-blocking and synchronised checkpointing. The log based maintains log of data regarding the consistent state, log based can be classified as pessimistic, optimistic and causal.

The goals of this work are as follows:

- To design a WSN that is fault tolerable by implementing checkpointing and rollback recovery.
- In fusing coordinated and non blocking checkpointing thereby making the system a well orchestrated one.
- Implementing demand checkpointing as a result saving time and energy.

1.2 COLLECTION TREE PROTOCOL

The Collection Tree Protocol (CTP) [4] is a routing protocol for wireless sensor networks. It is used for transferring data from one or more sensors to one or more root nodes. The number of Expected Transmissions (ETX) needed to send data between two nodes is used as the routing metric. Routes with lower metric are preferred. In a route that includes multiple hops, the metric is the sum of the ETX of the individual hops. Each node that wishes to collect data advertises itself as a tree root. Each node sends its data to the tree root to which it is nearest, that is, the tree root from which it is separated by the smallest ETX. A tree root always has an ETX of zero. Each node only keeps the smallest ETX to the nearest tree root. The grouping of ETX values is known as a gradient, and messages are propagated only from nodes with higher ETX to nodes with smaller ETX. This kind of forwarding is common in wireless sensor networks.

CTP tries to avoid routing loops by having a node transmit a beacon frame if it receives a packet with ETX lower than its own value. The aim is to have the sender of the packet receive the beacon frame and adjust its ETX accordingly. CTP should choose the route with the lowest ETX, the equations for calculating the ETX are in the Eq.(1) and Eq.(2):

$$ETX_{root} = 0 \qquad (1)$$

$$ETX_{node} = ETX_{parent} + ETX_{link\_to\_parent} \qquad (2)$$

CTP addresses loops using certain mechanism. First, every CTP packet contains a node's current gradient value. If CTP receives a data frame with a gradient value lower than its own, then this indicates that there is an inconsistency in the tree. CTP tries to resolve the inconsistency by broadcasting a beacon frame, with an expectation that the node which sent the data frame will hear it and adjust its routes accordingly.

Section 2 discusses works related to fault tolerance in WSN. Section 3 explains the proposed system. Section 4 shows the simulation and performance analysis of the proposed system. Finally we conclude the paper in section 5.

2. RELATED WORK

To make a system fault tolerable few popular approaches are dealt [3], this involves installation of additional node to nodes in the environment for monitoring or to repair the nodes that had failed to do the assigned work, another approach is to develop a protocol for attaining the *K* coverage or connectivity of the nodes. These approaches incur high cost and high power consumption.

In general, checkpointing techniques can be classified into two categories [3]:

- First approach is the latency hiding technique that tends to hide the execution of checkpointing from application programs. Well-known examples are forked checkpointing and copy-on-write checkpointing.
- Second approach is the size reduction technique that aims to minimize the amount of checkpoint data thereby reducing cost. Examples include incremental check-pointing and memory exclusion checkpointing. Examples include incremental check-pointing and memory exclusion checkpointing

Further the checkpoint implemented is not coordinated among the nodes and therefore checkpoint taken is not consistent. While taking checkpoint and rolling back to the error free state, the system will be in a blocked state i.e. other works running behind will be stopped until the complete process is over.

A new approach was proposed [3] called incremental checkpointing, where the redundancy of the data stored in memory are eliminated thereby saving the memory space, incremental checkpointing is a two step process where in first step full checkpoint is taken, where error free system is snap shot and stored in memory. In second step each time while taking checkpoint, the check pointed data is compared with the full checkpoint data and varying parameter is only stored in the memory there by reducing the redundancy, overhead and minimizing the space required to save the data. After the first full Checkpointing the following checkpoints are called as incremental. The main drawback of this paper is the dual mote architecture involved in designing the system, in dual mote architecture for every primary node deployed in the system there needs an additional node which is called as back up node where the checkpoint file are stored and roll backed in faulty conditions, this increases the cost for achieving the fault tolerant state and this indeed increases the overhead involved while checkpointing.

Checkpointing can be network-wide [5]: all nodes synchronously store local checkpoints to external flash. Network-wide checkpointing can be configured to trigger periodically. Checkpointing can also be on-demand: an external observer sends a checkpoint command to a specific node, triggering a node local checkpoint that the observer may download. Sensornet checkpointing consists of two network-wide operations: checkpoint and rollback, and is inspired by the substantial body of work in distributed checkpointing. The disadvantage of this paper is network execution is fully frozen as seen in Fig.2 while checkpoints are taken this interrupts the normal working of the node. So the greatest challenge is to overcome this frozen state.

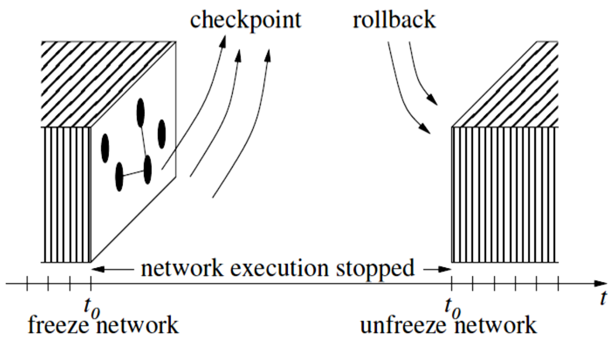


Fig.2. Frozen network while taking checkpoint

The energy [6] consumption of Ubiquitous Sensor Networks (USN) must be considered since sensor node inter-work with one another on USNs to transfer data to the sink node, the lifetime of each sensor becomes an important factor for determining that of the USN. Much energy is used in the following process: Initialization process where clusters are built, the normal operations of collecting data and transferring them to the sink node, or while the routing table is updated when an error occurs on a node during normal operation. Checkpointing and rollback recovery is used here. The task previously performed runs again after the reset. In this case, the routing table saved in the flash memory is copied into the memory of the new task. The task is preserved by using the latest routing table. The usage of watch dog timer is a new idea and also since many nodes deployed will have an inbuilt watch dog timer it can be used effectively. Overhead for recovery is determined by the energy required for sending messages and a time required to build a routing table. There are four extensions to sensornet checkpointing compression [7], binary diffs, selective checkpointing, and checkpoint inspection that reduce the time required for checkpointing operations considerably, and improve the granularity at which system state can be examined and manipulated down to the variable level.

3. PROPOSED SYSTEM

The basic architecture of the system proposed is shown in Fig.3, where we have set of nodes, each node as its own property and working. The nodes in Fig.3 are normal node which senses some value and forward it, next an important node called cluster head which act like a base station collect and stores the sensed information from the normal node, these cluster head have abundant amount of power. In addition to above said nodes a backup node is deployed in the system which acts like a standby to cluster node at the time of fault occurrence. The communication link between the nodes and sink node is established with a help of special protocol called collection tree protocol (CTP), designed exclusively for WSN.

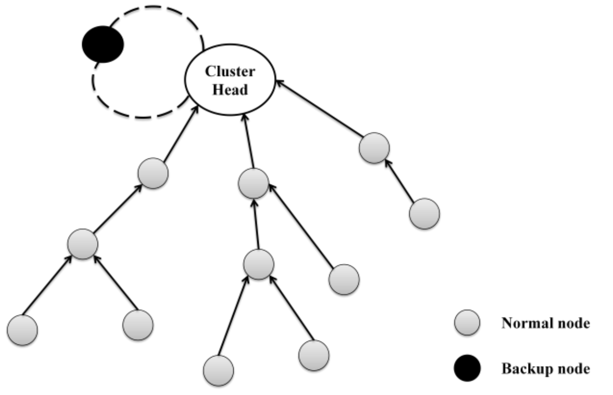


Fig.3. Tree structure

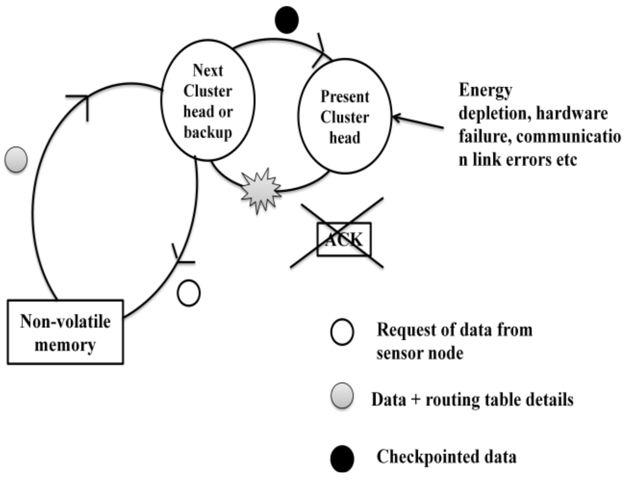

Next Cluster head or backup
Present Cluster head
Energy depletion, hardware failure, communicatio n link errors etc
ACK
Non-volatile memory
Request of data from sensor node
Data + routing table details
Checkpointed data

Fig.4. Working of proposed system

The Fig.4 describes the working of designed checkpointing scheme at the presence of fault in a cluster; at the time of fault occurrence the checkpointed data stored in the backup node is rolledback or recovered from next cluster head or backup node for proper working of the network, there by achieving fault tolerance in a designed network. The checkpointed data avoids the system to be restarted from the initial stage, instead the backup node will take the role of cluster head and starts from the best known error free state.

The working flow of the proposed system as follows:

1) Cluster head send initiator message to all nodes in the group, Nodes which are ready, send ready message to cluster head, Tree structure is formed with the help of CTP protocol.
2) The backup node will take snap shot (i.e. checkpoints) of the current working state and saves in the memory.
3) If the cluster head crashes, to bring back it to normal working state the saved checkpointed data in the memory will be rolledback.
4) Now the crashed system will work normally from the most recent error free state.

The main advantage of our proposed system is while taking checkpoint the back end process run normally; the system will not be in frozen state until the data is checkpointed.

4. SIMULATION AND PERFORMANCE ANALYSIS

The simulation of the proposed system is carried in Cooja simulator present in Contiki OS. Cooja Simulator [9] is a network simulator specifically designed for Wireless Sensor Networks. The Contiki Shell is an interactive on-mote UNIX-style shell that allows for text-based interaction with a sensor node or a network of sensor nodes through a set of commands that can be executed on a UNIX like command line terminal. It has features such as piping data, run in background, file system interaction, network commands, sensor measurement commands and system commands. The shell can be accessed either over a serial USB connection or over a network using Telnet. Shell commands are used to perform checkpoint and rollback recovery. The commands used in shell are:

1) For checkpointing “checkpoint <filename>”;

2) For rollback recovery “rollback <filename>”.

CTP is analyzed and compared with on demand checkpointing for various numbers of nodes, parameters like packet delivery ratio and time delay is calculated. Four scenarios are taken with number of nodes as 10, 20, 30, and 40. The parameters are given as shown in Table.1. The simulation time is set as 5 minutes and seed value is 1000 for all scenarios. The time required for checkpointing and rollback is measured from loglistener and graph is plotted.

Table.1. Simulation Parameters

Scenario	Property
No. of Nodes	10, 20, 30, 40
No. of faulty node	1, 2, 3, 4
Faulty node number	3, 6, 9, 12
Simulation Run Time	500 sec
MAC	CSMA CONTIKI MAC
SEED	1000
ROUTING PROTOCOL	CTP & DEMAND CHECKPOINT
Transmission Range	100m
MOTE TYPE	SKY

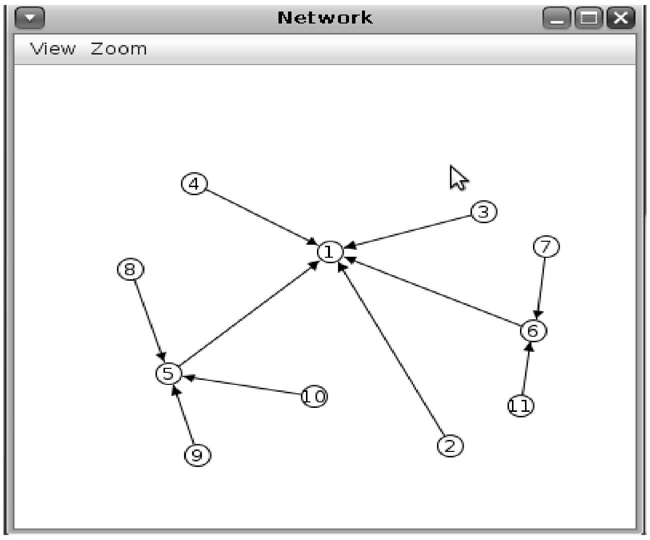
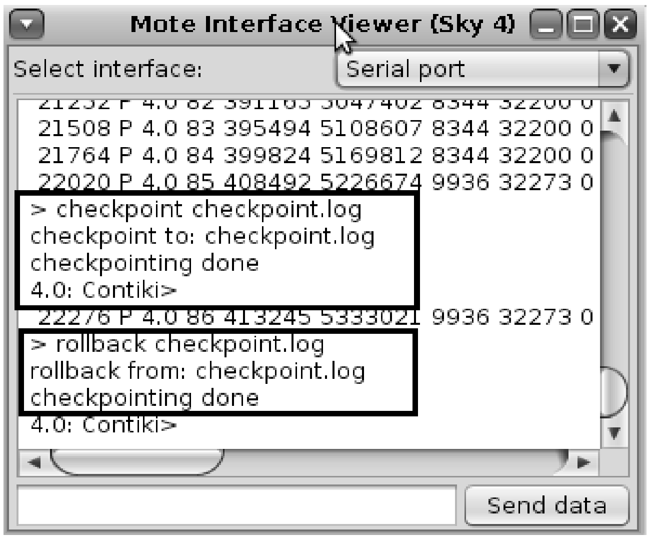


Fig.5. Tree formation among 10 nodes

The tree formation among 10 nodes is described in Fig.5, the tree is formed using CTP protocol, here node 1 act as cluster head and node 4 act as backup node, and in node 4 we are going to take checkpoint and rollback using commands. Node 3 act as an faulty node in the network. The number of faulty node increases as number of node increases. The faulty node number will be in multiples of 3 i.e. for 10 nodes scenario node 3 is faulty, for 20 nodes scenario node 3, 6 are faulty, for 30 node scenario node 3,6,9 are faulty and for 40 node scenario nodes 3, 6, 9, 12 are faulty nodes. The nature of faulty node is, it will not forward any packet to its cluster head, the entire packet or message it receives will be dropped without any prior intimation to the sender.



```
 21252 P 4.0 82 391163 5047402 8344 32200 0
 21508 P 4.0 83 395494 5108607 8344 32200 0
 21764 P 4.0 84 399824 5169812 8344 32200 0
 22020 P 4.0 85 408492 5226674 9936 32273 0
> checkpoint checkpoint.log
checkpoint to: checkpoint.log
checkpointing done
4.0: Contiki>
 22276 P 4.0 86 413245 5333021 9936 32273 0
> rollback checkpoint.log
rollback from: checkpoint.log
checkpointing done
4.0: Contiki>
```

Fig.6. Mote interface viewer

Time Delay: The time delay for sending hello message from various nodes to sink is calculated, and tabulated as shown in Fig.7 it was found that there is slight variation in time delay between CTP, CTP infused with demand checkpointing and checkpoint with faulty node. The time delay in millisecond (ms) is taken as Y axis and number of nodes is taken as X axis. The variation is caused due to the implementation of checkpointing concept in the network. The delay increases as the number of node increases. The Fig.6 shows checkpointing and rollback recovery process on mote interface viewer of Contiki OS.

Packet Delivery Ratio (PDR): PDR is defined as the ratio between total number of packet received by total number of packet transmitted. PDR is calculated by sending data i.e. Hello message with same sequence number from different nodes to sink, the sink is monitored for received message with the same sequence number. The graph is drawn with CTP, CTP infused with checkpoint and CTP with checkpoint including faulty node.

The PDR value in percentage is taken as Y axis and number of nodes is taken as X axis. The PDR value decreases gradually as number of node increases as shown in Fig.8.

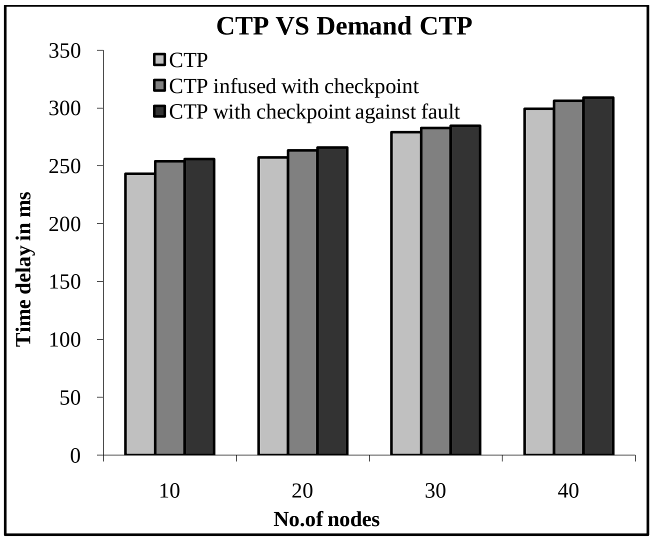


Fig.7. Time delay for sending Hello message from nodes to sink

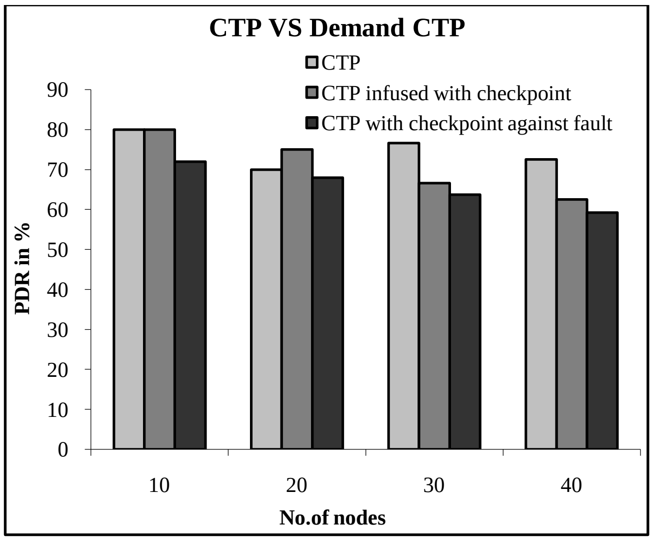


Fig.8. Packet Delivery Ratio of CTP and Demand CTP

The time involved to take checkpointing and to rollback is measured and tabulated in the Table.2.

Table.2. Time involved in taking Checkpoint

STATE	TIME (in Milli Seconds)
Checkpointing Time	10695
Rollback Time	1572

The file size of the file before and after taking checkpoint is tabulated in the Table.3, this shows the file size reduction should be done to reduce the over head of the file saved.

Table.3. File Size of the data

STATE	FILE SIZE (in bytes)
Before Checkpoint	60
After Checkpoint	682

5. CONCLUSION AND FUTURE WORK

Thus on-demand checkpointing and rollback recovery are done to achieve fault tolerance in a wireless sensor network. The checkpointing done here is a non blocking type where the system will be in unfrozen state rather than traditional frozen state seen earlier. Various scenarios have been evaluated and tabulated by increasing number of nodes. From Fig.7, the graph drawn with time delay as parameter shows that the time delay for demand CTP infused with checkpoint is slightly high compared with CTP, with slight sacrifice in time delay we achieved improved packet transmission during fault occurrence. Another reason for the reduced performance in the system is, here we are considering only one node as a backup node for all 4 scenarios. Increasing the backup nodes count could lead to improved performance. As a part of future work, we will provide a mechanism to reduce the size of checkpointed file and also, we have planned to design an optimized and enhanced protocol for network wide checkpointing which is to be dynamic in nature for achieving fault tolerance in the WSN.

REFERENCES

[1] Holger Karl and Andreas Willig, "*Protocols and Architectures for Wireless Sensor Networks*", John Wiley & Sons, Ltd, 2005.

[2] Arunanshu Mahapatro and Pabitra Mohan Khilar, "On distributed self fault diagnosis for wireless multimedia sensor networks", *International Conference on Communication, Computing and Security*, pp. 100-105, 2011.

[3] Hsung-Pin Chang, Yen-Ting Liu and Shang-Sheng Yang, "Surviving sensor node failures by MMU-less incremental checkpointing", *Journal of Systems and Software- Elsevier*, Vol. 87, pp. 74-86, 2014.

[4] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss and Philip Levis, "Collection tree protocol", *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pp. 1-4, 2009.

[5] Fredrik Österlind, Adam Dunkels, Nicolas Tsiftes and Zhitao He, "Sensornet Checkpointing between Simulated and Deployed Networks", *International Conference on Information Processing in Sensor Networks*, pp. 411-412, 2009.

[6] Kwang-Mo Jung, Joong-Jin Kook, Kwang-Soon Choi, Seong-Dong Kim and SangWon Min, "Energy Efficient Route Recovery Methods for Wireless Sensor Networks Using Hybrid Checkpointing", *Computational Science and Its Applications-Lecture Notes in Computer Science*, Vol. 4706, pp. 593-601, 2007.

[7] Andreas Loscher, Nicolas Tsiftes, Thiemo Voigt and Vlado Handziski, "Efficient and Flexible Sensornet Checkpointing", *Wireless Sensor Network, Lecture Notes in Computer Science-Springer International publishing*, Vol. 8354, pp. 50-65, 2014.

[8] Mukesh Singhal, and Niranjan G. Shivaratri, "*Advanced Concepts In Operating Systems*", Tata McGraw-Hill, Inc., 2008.

[9] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne and T. Voigt, "Cross-level sensor network simulation with COOJA", *Proceedings of 31st IEEE Conference on Local Computer Networks*, pp. 641-648, 2006.